

PARALLEL PROGRAMMING STRATEGIES FOR IRREGULAR ADAPTIVE APPLICATIONS

R. Biswas

NASA Ames Research Center
Moffett Field, CA 94035-1000
rbiswas@nas.nasa.gov

Achieving scalable performance for dynamic irregular applications is eminently challenging [1]. Traditional message-passing approaches have been making steady progress towards this goal; however, they suffer from complex implementation requirements. The use of a global address space greatly simplifies the programming task, but can degrade the performance for such computations. In this work, we examine two typical irregular adaptive applications, Dynamic Remeshing and N-Body, under competing programming methodologies and across various parallel architectures [2]. The Dynamic Remeshing application simulates flow over an airfoil, and refines localized regions of the underlying unstructured mesh. The N-Body experiment models two neighboring Plummer galaxies that are about to undergo a merger. Both problems demonstrate dramatic changes in processor workloads and interprocessor communication with time; thus, dynamic load balancing is a required component.

We investigate several critical factors of the parallel code development, including performance, programmability, scalability, algorithmic features, and portability. We compare the most popular implementation strategy, message passing with MPI, against a number of alternate approaches. First, we evaluate the effectiveness of using the SHMEM communication library, which uses symmetric address spaces for individual processes, allowing one-sided communication. Next, we examine an implementation using the OMP programming strategy, which uses shared-memory algorithms and OpenMP-style compiler directives on systems supporting global addressing. The cache-coherent shared address space (CC-SAS) paradigm is tested next. It is similar to OMP but focuses on spatial locality through methods such as data remapping and replication, which are traditionally not considered in shared-memory programming. Experimental results using hybrid programming are then reported. Here, two layers of parallelism are combined by implementing OpenMP codes within shared-memory multiprocessors (SMPs), while using message passing between the SMP nodes. Finally, we present results on the Cray MTA that uses multithreading to hide latency, rather than using data caches.

Experiments were performed on several parallel systems: the distributed-memory Cray T3E, the hardware-supported CC-NUMA SGI Origin2000, the latest generation of IBM POWER3 8-way SP cluster, and the Cray MTA multithreaded computer. Results indicate that it is possible to achieve message-passing performance using shared-memory programming techniques by carefully following the same high-level strategies, and that multithreaded systems offer tremendous potential but are not well suited for all classes of applications.

References

- [1] L. Oliker and R. Biswas, "Parallelization of a Dynamic Unstructured Algorithm Using Three Leading Programming Paradigms," *IEEE Transactions on Parallel and Distributed Systems*, v. 11, p. 931-940, 2000.
- [2] H. Shan, J.P. Singh, L. Oliker, and R. Biswas, "A Comparison of Three Programming Models for Adaptive Applications on the Origin2000," *CD-ROM Proceedings of SC2000 Conference*, Dallas, TX, 2000.